```python
import numpy as np
import cv2
import pandas as pd
import matplotlib.pyplot as plt
from collections import Counter
from google.colab.patches import cv2_imshow
#finding Primary Direction
def findMax(a,b,c,d):
  primaryDirection=-1
  maxi=max([a,b,c,d],key=abs)
  if maxi==a:
    primaryDirection=0
  elif maxi==b:
    primaryDirection=1
  elif maxi==c:
    primaryDirection=2
  else:
    primaryDirection=3

  return primaryDirection,maxi

# finding Secondary Direction

def findSecondMax(a,b,c,d):
  secondaryDirection=-1
  l=[a,b,c,d]

  maxi=max(l,key=abs)
  l.remove(maxi)
  maxi=max(l,key=abs)
  if maxi==a:
    secondaryDirection=0
  elif maxi==b:
    secondaryDirection=1
  elif maxi==c:
    secondaryDirection=2
  else:
    secondaryDirection=3

  return secondaryDirection,maxi
# finding Ternary Pattern
def findTernary(value,sigma):
  if value<-sigma:
    return 2
  elif value>sigma:
    return 1
  elif -sigma<=value and value<=sigma:
    return 0
  else:
    pass
#function to find the thresold value
from deepface import DeepFace as KNN_CLASSIFIER
def findSigma(matrix):
  matrix.sort()
  return sum(matrix)//len(matrix)
# finding ldtp codes
def findLDTP(d1,d2,t1,t2):
  res=0
  res+=(2**6)*d1
  res+=(2**4)*t1
  res+=(2**2)*d2
  res+=t2
  return res
# K-NN Classifier

def KNN_CLASSIFIR(data_set):

  #Extracting Independent and dependent Variable
  x= data_set.iloc[:,1:].values
  y= data_set.iloc[:,0]

  #Splitting the dataset into training and test set.
  from sklearn.model_selection import train_test_split
  x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 0
.25, random_state=0)

  #feature Scaling
  from sklearn.preprocessing import StandardScaler
  st_x= StandardScaler()
  x_train= st_x.fit_transform(x_train)
  x_test= st_x.transform(x_test)
def findEmotion(image):
#Masks M-0 to M-3
##Mask
  M=[
    [
      [-1,0,1],
      [-2,0,2],
      [-1,0,1]],
    [
      [0,1,2],
      [-1,0,1],
      [-2,-1,0]],
    [
      [1,2,1],
      [0,0,0],
    [-1,-2,-1] ],
  [
    [2,1,0],
    [1,0,-1],
    [0,-1,-2]
  ]
]

temp=[[0,0,0],[0,0,0],[0,0,0]]

r,c= image.shape
LDTP = np.zeros((r-2,c-2))

#intialize all the lists to empty

matrix1=[]
matrix2=[]
matrix3=[]
matrix4=[]



for i in range(r-2):
  for j in range(c-2):
    for k in range(3):
      for l in range(3):
        temp[k][l]=image[i+k][j+l]

    #convolution with first mask
    res1=0
    s=M[0]
    for k in range(3):
      for l in range(3):
        matrix1.append(s[k][l]*temp[k][l])
        res1+=s[k][l]*temp[k][l]


    #convolution with second mask
    res2=0
    s=M[1]
    for k in range(3):
      for l in range(3):
        matrix2.append(s[k][l]*temp[k][l])
        res2+=s[k][l]*temp[k][l]


    #convolution with third mask
    res3=0
    s=M[2]
```

```
for k in range(3):
  for l in range(3):
    matrix3.append(s[k][l]*temp[k][l])

    res3+=s[k][l]*temp[k][l]


#convolution with fourth mask
res4=0
s=M[3]
for k in range(3):
  for l in range(3):
    matrix4.append(s[k][l]*temp[k][l])
    res4+=s[k][l]*temp[k][l]


primaryDirection,maximumValue=findMax(res1,res2,res3,res4)

secondaryDirection,secondMax=findSecondMax(res1,res2,res3,res4)

if primaryDirection==0:

  sigma=findSigma(matrix1)
elif primaryDirection==1:

  sigma=findSigma(matrix2)
elif primaryDirection==2:

  sigma=findSigma(matrix3)
elif primaryDirection==3:

  sigma=findSigma(matrix4)

if secondaryDirection==0:
  sigma1=findSigma(matrix1)
elif secondaryDirection==1:
  sigma1=findSigma(matrix2)
elif secondaryDirection==2:
  sigma1=findSigma(matrix3)
elif secondaryDirection==3:
  sigma1=findSigma(matrix4)



primaryTernary=findTernary(maximumValue,sigma)

secondaryTernary=findTernary(secondMax,sigma1)
```

```
      LDTP[i][j]=findLDTP(primaryDirection,secondaryDirection,primaryTernary,secondaryTernary)

      matrix1=[]
      matrix2=[]
      matrix3=[]
      matrix4=[]


  #printing the resultant LDTP for the iamge

  print(LDTP)

  #converting the LDTP to histograms

  plt.hist(LDTP.ravel(),256,[0,256])
  print("The histogram generated based on the ldtp codes is")
  print()
  plt.show()
  return LDTP
img = cv2.imread("Happy.jpg")
image=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
LDTP=findEmotion(image)
data_set= pd.read_excel('Final Data.xlsx')

def testFace(data_set,img):

  result = KNN_CLASSIFIER.analyze(img,actions = ['emotion'])

  emotions_dict = result['emotion']
  Keymax = max(zip(emotions_dict.values(), emotions_dict.keys()))[1]

  print("Detected Emotion is:",Keymax)


testFace(data_set,img)
image=cv2.resize(image,(48,48),interpolation= cv2.INTER_LINEAR)
cv2_imshow(image)
```